Data Science e Tecnologie per le Basi di Dati

Esercitazione di laboratorio n. 1

Data warehouse: SQL esteso

La finalità di questa esercitazione consiste nella realizzazione di un data warehouse conforme alle specifiche riportate nei punti seguenti, utilizzando Oracle. Oltre alla progettazione del data warehouse, sarà necessario risolvere alcune interrogazioni in SQL esteso.

La traccia dell'esercitazione è la seguente:

1. Descrizione del problema
2. Descrizione della base dati OLTP
3. **Esercizio**: progettazione del data warehouse
4. **Esercizio**: confronto con lo schema logico del data warehouse
5. **Esercizio** (SQL Developer): interrogazione del data warehouse

1. Descrizione del problema

Una società di telefonia mobile è interessata ad analizzare i dati a sua disposizione per fornire un servizio mirato ai suoi clienti e per migliorare la distribuzione delle proprie apparecchiature sul territorio. Attualmente la società telefonica dispone di basi di dati contenenti tutte le informazioni relative alle chiamate effettuate dai suoi utenti. In particolare, per ogni chiamata sono noti il numero di telefono del chiamante, il numero di telefono del chiamato, la durata della telefonata, la tariffa applicata e l'istante di inizio della chiamata (data, ora, minuto, secondi).

La dirigenza della società vuole poter ottenere velocemente delle informazioni sul traffico telefonico effettuato sulle linee telefoniche dell'azienda e sui guadagni effettuati su base giornaliera in funzione della località del chiamante, del giorno e della tariffa.

In particolare, alcune delle informazioni a cui i dirigenti sono interessati sono:

- Incassi e numero di chiamate effettuate su base mensile in funzione della città nella quale si trova l'apparecchio del chiamante
- Incassi e numero di chiamate effettuate su base mensile in funzione della città nella quale si trova l'apparecchio del chiamato
- Incassi e numero di chiamate effettuate su base mensile in funzione della provincia e della regione nella quale si trova l'apparecchio del chiamante
- Incassi e numero di chiamate effettuate su base mensile in funzione della provincia e della regione nella quale si trova l'apparecchio del chiamato
- Incassi e numero di chiamate effettuate in base alla data nella quale si effettua la chiamata e della provincia nella quale si trova l'apparecchio del chiamante
- Incassi e numero di chiamate effettuate su base annua in funzione della provincia e della regione nella quale si trova l'apparecchio del chiamante
- Incassi e numero di chiamate effettuate su base mensile in funzione della tipologia di tariffa utilizzata
- Incassi e numero di chiamate effettuate in funzione del giorno della settimana e della tipologia di tariffa utilizzata
- Numero di chiamate effettuate in funzione della data e della regione nella quale si trova l'apparecchio del chiamante
- Numero di chiamate effettuate in funzione della data e della regione nella quale si trova l'apparecchio del chiamato

2. Descrizione della base dati OLTP

La base di dati (OLTP) della società telefonica in cui vengono memorizzate le singole telefonate è riportata in Figura 1.

Tabelle	Descrizione
TARIFFE (TipologiaTariffa INT NOT NULL, NomeTariffa VARCHAR(20) NOT NULL, CostoAlSecondo FLOAT NOT NULL, PRIMARY KEY(TipologiaTariffa));	Tipologie (categorie) di tariffe esistenti Cardinalità: 7 tuple
LOCALITA (CodLocalita INT NOT NULL, Citta VARCHAR(20) NOT NULL, Provincia VARCHAR(20) NOT NULL, Regione VARCHAR(20) NOT NULL, PRIMARY KEY(CodLocalita));	Località Cardinalità: 1500 tuple
TELEFONATE (TelChiamante VARCHAR(20) NOT NULL, TelChiamato VARCHAR(20) NOT NULL, LocalitaChiamante INT NOT NULL, LocalitaChiamato INT NOT NULL, Data DATE NOT NULL, Ora INT NOT NULL, Minuti INT NOT NULL, Secondi INT NOT NULL, DurataTelefonataSecondi FLOAT NOT NULL, TipologiaTariffa INT NOT NULL, PRIMARY KEY(TelChiamante,TelChiamato,Data,Ora,Minuti,Secondi), FOREIGN KEY(TipologiaTariffa) REFERENCES TARIFFE(TipologiaTariffa) , FOREIGN KEY(LocalitaChiamante) REFERENCES LOCALITA(CodLocalita), FOREIGN KEY(LocalitaChiamato) REFERENCES LOCALITA(CodLocalita));	Telefonate effettute negli anni 2003, 2004 Cardinalità: 422416 tuple

Figura 1 – Base di dati sorgente contenente le informazioni sulle singole telefonate

3. Esercizio: progettazione del data warehouse

Progettare lo schema concettuale di un data warehouse per la gestione delle problematiche illustrate in precedenza. Lo schema progettato deve essere pensato in modo da consentire:

- Le analisi richieste dalla società di telefonia mobile
- La fase ETL (extraction, transformation, loading) per importare i dati dalla base OLTP (Figura 1) al data warehouse.

4. Esercizio: confronto con lo schema logico del data warehouse

Confrontare lo schema concettuale progettato nell'esercizio precedente con lo schema logico proposto in Figura 2. Controllare in particolare di aver scelto correttamente le **misure** ed il **livello di aggregazione** dei dati. Osservando lo schema logico in Figura 2, rispondere alle seguenti domande:

- Quali sono le **misure** scelte per il data warehouse?
- Qual è **il livello di aggregazione** minimo nel data warehouse? Corrisponde con quanto progettato nello schema concettuale?

Tabelle			**Descrizione**
DWABD.TEMPO (ID_TEMPO DATA GIORNO MESE ANNO **PRIMARY** KEY(ID_TEMPO));	 NUMBER DATE VARCHAR VARCHAR NUMBER	 NOT NULL, NOT NULL, NOT NULL, NOT NULL, NOT NULL,	Dimensione tempo Cardinalità: 30 tuple
DWABD.TARIFFA (ID_TAR TIPO_TARIFFA **PRIMARY** KEY(ID_TAR));	 NUMBER VARCHAR	 NOT NULL, NOT NULL,	Dimensione tariffa Cardinalità: 7 tuple
DWABD.LUOGO (ID_LUOGO CITTA PROVINCIA REGIONE **PRIMARY** KEY(ID_LUOGO));	 NUMBER VARCHAR VARCHAR VARCHAR	 NOT NULL, NOT NULL, NOT NULL, NOT NULL,	Dimensione luogo (località) Cardinalità: 1500 tuple
DWABD.FATTI (ID_TEMPO ID_TAR ID_LUOGO_CHIAMANTE ID_LUOGO_CHIAMATO PREZZO CHIAMATE	 NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER	 NOT NULL, NOT NULL, NOT NULL, NOT NULL, NOT NULL, NOT NULL,	Tabella dei fatti Cardinalità: 7809 tuple
PRIMARY KEY(ID_TEMPO, ID_TAR,ID_LUOGO_CHIAMANTE,ID_LUOGO_CHIAMATO), FOREIGN KEY(ID_TEMPO) REFERENCES TEMPO(ID_TEMPO), FOREIGN KEY(ID_TAR) REFERENCES TARIFFA(ID_TAR), FOREIGN KEY(ID_LUOGO_CHIAMANTE) REFERENCES LUOGO(ID_LUOGO), FOREIGN KEY(ID_LUOGO_CHIAMATO) REFERENCES LUOGO(ID_LUOGO));			

- **Figura 2 – Tabelle del data warehouse**

5. Esercizio: interrogazione del data warehouse

5.1 Importare le tabelle in Sql Developer

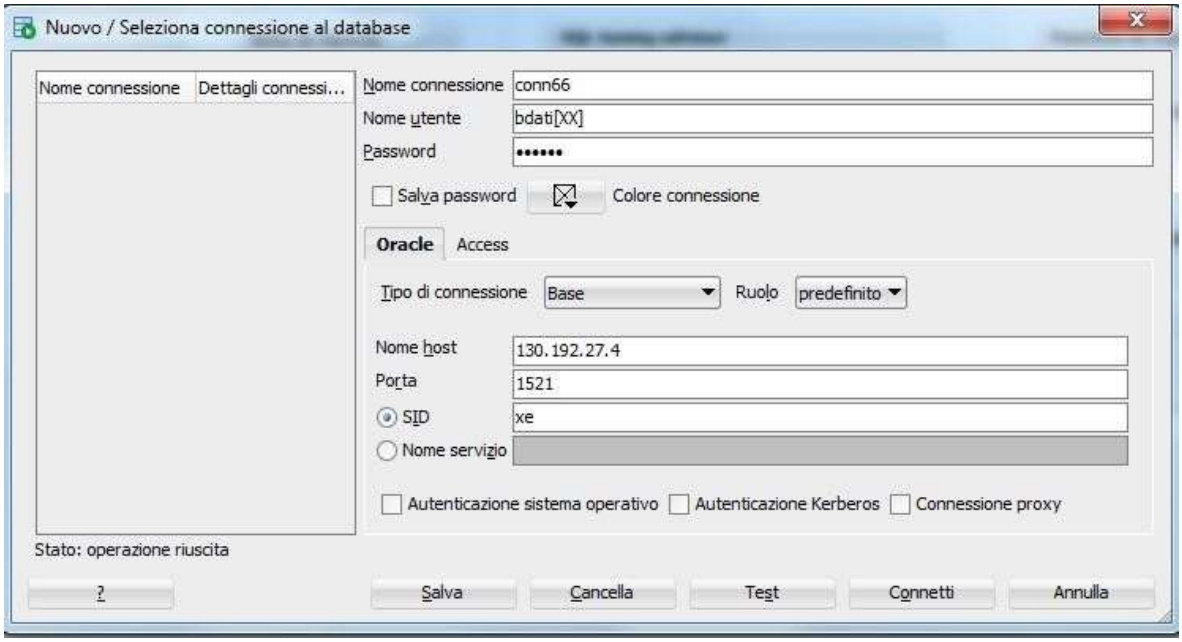
Aprire il programma Oracle SQL Developer
Cliccare su crea nuova connessione:

5.2 Connessione al database Oracle

Per autenticarsi inserire i seguenti parametri.

- Nome utente: bdati[scegliere un valore compreso tra 1-100]
- Password: orac[scegliere un valore compreso tra 1-100]
- Nome host: 130.192.27.4
- Porta: 1521
- SID: xe

Ad esempio, collegandosi dalla macchina numero 23 del laboratorio, usare come username **`bdati23`** e come password **`orac23`**.



5.3 Importare i dati sul proprio database

Per proseguire l'esercitazione è necessario disporre delle tabelle del data warehouse sul proprio database.
I dati da importare sono disponibili sul sito web del corso, vicino al testo di questa esercitazione.

- Scaricare i dati delle tabelle sul proprio pc. I nomi dei files sono: fatti.csv, luogo.csv, tariffa.csv, tempo.csv
- Assicurarsi che il proprio database non contenga già le tabelle (FATTI, LUOGO, TARIFFA, TEMPO). Nel caso esistano già è necessario rimuoverle (eseguire le query "DROP TABLE" oppure cancellare le tabelle dal menu):
    - DROP TABLE DWABD.FATTI;
    - Si noti l'utilizzo del **prefisso DWABD** per accedere correttamente ai nomi delle tabelle nel database di Laboratorio.
- Importare i **file csv (comma separated values)** nel database (tasto destro sulla voce "Tabelle" nel menù).
- Prestare attenzione ai **tipi di dato** associati ad ogni colonna durante la fase di import. In particolare assicurarsi che il campo DATA nella tabella TEMPO sia interpretato come *Date*.

5.4 Query in linguaggio SQL esteso

Risolvere usando il linguaggio SQL (versione estesa) le seguenti interrogazioni:

1. Selezionare l'incasso totale per ogni tipo tariffa e per ogni anno. Selezionare inoltre l'incasso totale complessivo, l'incasso totale per ogni tipo di tariffa indifferentemente dall'anno, e l'incasso totale per ogni anno indifferentemente dal tipo di tariffa.

2. Per ogni mese, selezionare il numero di chiamate totali e l'incasso totale. Utilizzando la funzione RANK(), associare ad ogni mese un numero che identifica la posizione del mese all'interno dei mesi in funzione dell'incasso totale effettuato (1 per il mese che ha incassato di più, 2 per il secondo mese, ecc.)

3. Selezionare per ogni mese dell'anno 2003 il numero di chiamate totali. Utilizzando la funzione RANK(), associare ad ogni mese un numero che identifica la posizione del mese all'interno dei vari mesi dell'anno 2003 in funzione del numero di chiamate totali (1 per il mese con più telefonate, 2 per il secondo mese, ecc.)

4. Selezionare per ogni data del mese di luglio 2003 l'incasso totale, e la media giornaliera degli incassi delle chiamate effettuate negli ultimi tre giorni.

5. Selezionare per ogni mese, l'incasso del mese e l'incasso cumulativo dall'inizio dell'anno.

6. Considerare l'anno 2003. Separatamente per tariffa e mese, analizzare (i) l'incasso totale, (ii) la percentuale dell'incasso rispetto all'incasso totale considerando tutte le tariffe telefoniche, (iii) la percentuale dell'incasso rispetto all'incasso totale considerando tutti i mesi.

7. Per ogni regione chiamante, selezionare il numero mensile di chiamate e il numero mensile cumulativo di chiamate dall'inizio dell'anno.

8. Considerare l'anno 2003. Analizzare l'incasso totale (i) separatamente per ogni mese, (ii) separatamente per ogni mese, tariffa telefonica e regione chiamante e (iii) separatamente per ogni mese, tariffa telefonica e regione ricevente.

```
SELECT SUM(PRICE), DATEYEAR, PHONERATETYPE
FROM FACTS F, TIMEDIM TE, PHONERATE TA
WHERE F.Id_TIME = TE.Id_TIME and F.Id_PHONERATE = TA.Id_PHONERATE
GROUP BY cube(PHONERATETYPE, DATEYEAR)
```

```
SELECT DATEYEAR, PHONERATETYPE, SUM(PRICE),
SUM(SUM(PRICE)) OVER (PARTITION BY PHONERATETYPE), SUM(SUM(PRICE)) OVER(PARTITION BY DATEYEAR),
SUM(SUM(PRICE)) OVER ()
FROM FACTS F, TIMEDIM TE, PHONERATE TA
WHERE F.Id_TIME = TE.Id_TIME and F.Id_PHONERATE = TA.Id_PHONERATE
GROUP BY PHONERATETYPE, DATEYEAR
```


```
SELECT SUM(PRICE), DATEYEAR, PHONERATETYPE
FROM FACTS F, TIMEDIM TE, PHONERATE TA
WHERE F.Id_TIME = TE.Id_TIME and F.Id_PHONERATE = TA.Id_PHONERATE
GROUP BY cube(PHONERATETYPE, DATEYEAR)
```

```
SELECT DATEYEAR, PHONERATETYPE, SUM(PRICE),
SUM(SUM(PRICE)) OVER (PARTITION BY PHONERATETYPE), SUM(SUM(PRICE)) OVER(PARTITION BY DATEYEAR),
SUM(SUM(PRICE)) OVER ()
FROM FACTS F, TIMEDIM TE, PHONERATE TA
WHERE F.Id_TIME = TE.Id_TIME and F.Id_PHONERATE = TA.Id_PHONERATE
GROUP BY cube(PHONERATETYPE, DATEYEAR)
```